Sliding Window Temporal Graph Coloring*

George B. Mertzios
Department of Computer Science
Durham University
Durham, UK
george.mertzios@durham.ac.uk

Hendrik Molter
Algorithmics and Computational Complexity, Fakultät IV, TU Berlin
Berlin, Germany
h.molter@tu-berlin.de

Viktor Zamaraev
Department of Computer Science
Durham University
Durham, UK
viktor.zamaraev@durham.ac.uk

Abstract

Graph coloring is one of the most famous computational problems with applications in a wide range of areas such as planning and scheduling, resource allocation, and pattern matching. So far coloring problems are mostly studied on static graphs, which often stand in stark contrast to practice where data is inherently dynamic and subject to discrete changes over time. A temporal graph is a graph whose edges are assigned a set of integer time labels, indicating at which discrete time steps the edge is active. In this paper we present a natural temporal extension of the classical graph coloring problem. Given a temporal graph and a natural number Δ, we ask for a coloring sequence for each vertex such that (i) in every sliding time window of Δ consecutive time steps, in which an edge is active, this edge is properly colored (i.e. its endpoints are assigned two different colors) at least once during that time window, and (ii) the total number of different colors is minimized. This sliding window temporal coloring problem abstractly captures many realistic graph coloring scenarios in which the underlying network changes over time, such as dynamically assigning communication channels to moving agents. We present a thorough investigation of the computational complexity of this temporal coloring problem. More specifically, we prove strong computational hardness results, complemented by efficient exact and approximation algorithms. Some of our algorithms are linear-time fixed-parameter tractable with respect to appropriate parameters, while others are asymptotically almost optimal under the Exponential Time Hypothesis (ETH).

1 Introduction

A great variety of modern, as well as of traditional networks are dynamic in nature as their link availability changes over time. Just a few indicative examples of such inherently dynamic networks are information and communication networks, social networks, transportation networks, and several physical systems (Holme and Saramäki 2013; Michail and Spirakis 2018). All these application areas share the common characteristic that the network structure, i.e. the underlying graph topology, is subject to *discrete changes over time*. In this paper, embarking from the foundational work of Kempe, Kleinberg, and Kumar (2002), we adopt a simple and natural model for time-varying networks, given by a graph with time-labels on its edges, while the vertex set is fixed.

Definition 1.1 (Temporal Graph)**.** *A* temporal graph *is a pair (G, λ), where $G = (V, E)$ is an underlying (static) graph and $\lambda : E \to 2^{\mathbb{N}}$ is a* time-labeling *function which assigns to every edge of G a set of discrete-time labels.*

For every edge $e \in E$ in the underlying graph G of a temporal graph (G, λ), $\lambda(e)$ denotes the set of time slots at which e is *active*. Due to their relevance and applicability in many areas, temporal graphs have been studied from various perspectives and under different names such as *time-varying* (Flocchini, Mans, and Santoro 2009; Tang et al. 2010; Aaron, Krizanc, and Meyerson 2014), *dynamic* (Casteigts et al. 2012; Giakkoupis, Sauerwald, and Stauffer 2014), *evolving* (Bui-Xuan, Ferreira, and Jarry 2003; Ferreira 2004; Clementi et al. 2010), and *graphs over time* (Leskovec, Kleinberg, and Faloutsos 2007). For a comprehensive overview on the existing models and results on temporal graphs from a (distributed) computing perspective see the surveys (Michail 2016; Latapy, Viard, and Magnien 2018; Casteigts et al. 2012; Casteigts and Flocchini 2013a; 2013b).

The conceptual shift from static to temporal graphs imposes new challenges in algorithmic computation and complexity. Now the classical computational problems have to be appropriately redefined in the temporal setting in order to properly capture the notion of time. Motivated by the fact that, due to causality, information in temporal graphs can "flow" only along sequences of edges whose time-labels are increasing, most temporal graph parameters and optimization problems that have been studied so far are based on the notion of temporal paths and other "path-related" notions, such as temporal analogues of distance, reachability, separators, diameter, exploration, and centrality (Akrida et al. 2016; Erlebach, Hoffmann, and Kammer 2015; Mertzios et al. 2013; Michail and Spirakis 2016; Akrida et al. 2017; Enright et al. 2018; Zschoche et al. 2018; Fluschnik et al. 2018).

Recently only few attempts have been made to define and study "non-path" temporal graph problems. Motivated by the contact patterns among high-school students, Viard, Latapy, and Magnien (2016), introduced Δ-cliques, an ex-

*GM and VZ are supported by the EPSRC grant EP/P020372/1. HM is supported by the DFG project MATE (NI 369/17).

tension of the concept of cliques to temporal graphs (see also (Himmel et al. 2017; Bentert et al. 2018)). Chen et al. (2018) presented an extension of the cluster editing problem to temporal graphs. Furthermore, Akrida et al. (2018) introduced the notion of temporal vertex cover, motivated by applications of covering problems in transportation and sensor networks. Temporal extensions of the classical graph coloring problem have also been previously studied by Yu et al. (2013) (see also (Ghosal and Ghosh 2015)) in the context of channel assignment in mobile wireless networks. In this problem, every edge has to be properly colored in every snapshot of the input temporal graph (G, λ), while the goal is to minimize some linear combination of the total number of colors used and the number of color re-assignments on the vertices (Yu et al. 2013). In this temporal coloring approach, the notion of time is only captured by the fact that the number of re-assignments affects the value of the target objective function, while the fundamental solution concept remains the same as in static graph coloring; that is, *every* individual (static) snapshot has to be properly colored. Using this, Yu et al. (2013) presented generic methods to adapt known algorithms and heuristics from static graph coloring to deal with their new objective function.

In this paper we introduce and rigorously study a different, yet natural temporal extension of the classical graph coloring problem, called SLIDING WINDOW TEMPORAL COLORING (for short, SW-TEMP. COLORING). In SW-TEMP. COLORING the input is a temporal graph (G, λ) and two natural numbers Δ and k. At every time step t, every vertex has to be assigned one color, under the following constraint: Every edge e has to be properly colored at least once during *every time window* of Δ consecutive time steps, and this must happen at a time step t in this window when e is active. Now the question is whether there exists such a temporal coloring over the whole lifetime of the input temporal graph that uses at most k colors. In contrast to the model of Yu et al. (2013), the solution concept in SW-TEMP. COLORING is fundamentally different to that of static graph coloring as it takes into account the inherent dynamic nature of the network. Indeed, even to verify whether a given solution is feasible, it is not sufficient to just consider every snapshot independently.

Our temporal extension of the static graph coloring problem is motivated by applications in mobile sensor networks and in planning. Consider the following scenario: every mobile agent broadcasts information over a specific communication channel while it listens on all *other* channels. Thus, whenever two mobile agents are sufficiently close, they can exchange information only if they broadcast on different channels. We assume that agents can switch channels at any time. To ensure a high degree of information exchange, it makes sense to find a schedule of assigning broadcasting channels to the agents over time which minimizes the number of necessary channels, while allowing each pair of agents to communicate at least once within every small time window in which they are close to each other.

To further motivate the questions raised in this work, imagine an organization which, in order to ensure compliance with the national laws and the institutional policies, requires its employees to *regularly* undertake special training that is relevant to their role within the organization. Such training requirements can be naturally grouped within training "themes", concerning –for example– the General Data Protection Regulation (GDPR) of the EU for staff dealing with personal data or equality and diversity issues when hiring new employees for Human Resources staff, etc).

One reasonable organizational requirement for such a regular staff training is that every employee has to undertake all needed pieces of training at least once within every time-window of a specific length Δ (e.g. $\Delta = 12$ months). All training sessions are offered by experts in predefined "training periods" (e.g. annually every January, May, and September), while each session takes a fixed amount of time to run (e.g. a full day during the corresponding training period). This situation can be naturally modeled as a *temporal* graph problem: (i) each time slot t represents a predefined "training period"; (ii) each vertex v denotes one of the themes that are offered for training by the organization; (iii) the different colors that a vertex v can take at time slot t represent all different days in which the theme v can be taught during the training period t; (iv) an edge $\{u, v\}$ that is active at the time slot t means that the themes u and v share at least one participant at the corresponding training period. Note that, since the training needs of specific staff members change over time, an edge between two themes u and v may repeatedly appear and disappear over time, and thus the above graph is temporal. If a participant is planned to undertake training on both themes u, v at the same time slot t, then these themes have to run at different days of the time slot t, i.e. u and v have to be assigned different colors at time t. In such a situation, it is natural for the organization to try to schedule all training sessions in such a way that the total *duration* (i.e. number of different colors) of every training period t never exceeds k different days, while simultaneously meeting all regular training requirements.

Our Contribution. In this paper we introduce the problem SLIDING WINDOW TEMPORAL COLORING (for short, SW-TEMP. COLORING) and we present a thorough investigation of its computational complexity. All our notation and the formal definition of the temporal problems that we study are presented in Section 2. First we investigate in Section 3 an interesting special case of SW-TEMP. COLORING, called TEMPORAL COLORING, where the length Δ of the sliding time window is equal to the whole lifetime T of the input temporal graph. We start by proving in Theorem 3.1 that TEMPORAL COLORING is NP-hard even for $k = 2$, and even when every time slot consists of one clique and isolated vertices. This is in wide contrast to the static coloring problem, where it can be decided in linear time whether a given (static) graph G is 2-colorable, i.e. whether G is bipartite. On the positive side, we show in Theorem 3.3 that, given any input temporal graph (G, λ) for TEMPORAL COLORING with n vertices and lifetime T, we can compute an equivalent instance (G', λ') on the same vertices but with lifetime $T' \leq m$, where m is the number of edges in the underlying graph G. Moreover we show that the new instance can be computed in polynomial time. Formally, The-

orem 3.3 shows that TEMPORAL COLORING admits a polynomial kernel when parameterized by the number n of vertices of the input temporal graph. That is, we can efficiently preprocess any instance of TEMPORAL COLORING to obtain an equivalent instance whose size only depends polynomially on the size of the underlying graph G and not on the lifetime T of (G, λ).

In Section 4 and in the reminder of the paper we deal with the general version of SW-TEMP. COLORING, where the value of Δ is arbitrary. On the one hand, we show that the problem is hard even on very restricted special classes of input temporal graphs. On the other hand, assuming the Exponential Time Hypothesis (ETH), we give an asymptotically optimal exponential-time algorithm for SW-TEMP. COLORING whenever Δ is constant. Moreover we show how to extend it to get an algorithm which runs in linear time if the number n of vertices is constant. Note here that the *size* of the input temporal graph also depends on its lifetime T whose value can still be arbitrarily large, independently of n. Furthermore note that this assumption about n being a constant can be also reasonable in practical situations; for example, in our motivation above about planning the training of staff in an organization, the value of n equals the number of different "training themes" to be run, which can be expected to be rather small.

Finally we consider in Section 4 an optimization variant of SW-TEMP. COLORING where the number of colors is to be minimized. We give an approximation algorithm with an additive error of 1 which runs in linear time on instances where the underlying graph G of the input temporal graph (G, λ) has a constant-size vertex cover. From a classification standpoint this is also optimal since the problem remains NP-hard to solve optimally on temporal graphs where the underlying graph has a constant-size vertex cover.

Due to space constraints, some proofs are deferred to a full version (Mertzios, Molter, and Zamaraev 2018).

2 Preliminaries and Notation

Given a (static) graph G, we denote by $V(G)$ and $E(G)$ the sets of its vertices and edges, respectively. An edge between two vertices u and v of G is denoted by $\{u, v\}$, and in this case u and v are said to be *adjacent* in G. A *complete graph* (or a *clique*) is a graph where every pair of vertices is adjacent. The complete graph on n vertices is denoted by K_n. For every $i, j \in \mathbb{N}$, where $i \leq j$, we let $[i, j] = \{i, i+1, \ldots, j\}$ and $[j] = [1, j]$. Throughout the paper we consider temporal graphs with *finite lifetime* T, that is, there is an maximum label assigned by λ to an edge of G, called the *lifetime* of (G, λ); it is denoted by $T(G, \lambda)$, or simply by T when no confusion arises. Formally, $T(G, \lambda) = \max\{t \in \lambda(e) : e \in E\}$. We refer to each integer $t \in [T]$ as a *time slot* of (G, λ). The *instance* (or *snapshot*) of (G, λ) *at time* t is the static graph $G_t = (V, E_t)$, where $E_t = \{e \in E : t \in \lambda(e)\}$. If $E_t = \emptyset$, we call $G_t = (V, E_t)$ a *trivial snapshot*. For every subset $S \subseteq [T]$ of time slots, we denote by $(G, \lambda)|_S$ the restriction of (G, λ) to the time slots in the set S. In particular, for the case where $S = [i, j]$ for some $i, j \in [T]$, where $i \leq j$, we have that $(G, \lambda)|_{[i,j]}$ is the sequence of the instances $G_i, G_{i+1}, \ldots, G_j$. We assume in the remainder of the paper that every edge of G appears in at least one time slot until T, namely $\bigcup_{t=1}^{T} E_t = E$.

In the remainder of the paper we denote by $n = |V|$ and $m = |E|$ the number of vertices and edges of the underlying graph G, respectively, unless otherwise stated. Furthermore, unless otherwise stated, we assume that the labeling λ is arbitrary, i.e. (G, λ) is given with an explicit list of labels for every edge. That is, the *size* of the input temporal graph (G, λ) is $O\left(|V| + \sum_{t=1}^{T} |E_t|\right) = O(n + mT)$. In other cases, where λ is more restricted, e.g. if λ is periodic or follows another specific temporal pattern, there may exist more succinct representations of the input temporal graph.

For every $v \in V$ and every time slot t, we denote the *appearance of vertex v at time t* by the pair (v, t). That is, every vertex v has T different appearances (one for each time slot) during the lifetime of (G, λ). For every time slot $t \in [T]$ we denote by $V_t = \{(v, t) : v \in V\}$ the set of all vertex appearances of (G, λ) at the time slot t. Note that the set of all vertex appearances in (G, λ) is the set $V \times [T] = \cup_{1 \leq t \leq T} V_t$.

TEMPORAL COLORING. A *temporal coloring* of a temporal graph (G, λ) is a function $\phi : V \times [T] \to \mathbb{N}$, which assigns to every vertex appearance (v, t) in (G, λ) one color $\phi(v, t) \in \mathbb{N}$. For every time slot $t \in [T]$ we denote by ϕ_t the restriction of ϕ to the vertex appearances at time slot t; then ϕ_t is referred to as the *time slot coloring* for the time slot t. That is, $\phi_t : V \to \mathbb{N}$, such that $\phi_t(v) = \phi(v, t)$, for every $v \in V$. Furthermore, for simplicity of the presentation, we will refer to the temporal coloring ϕ as the ordered sequence $(\phi_1, \phi_2, \ldots, \phi_T)$ of all its time slot colorings. Let $e \in E$ be an edge of the underlying graph G. We say that an edge $e = \{u, v\}$ of the underlying graph G is *temporally properly colored* at time slot t if (i) $\phi_t(u) \neq \phi_t(v)$, and (ii) $t \in \lambda(e)$, i.e. the edge e is *active* in the time slot t. We now introduce the notion of a *proper temporal coloring* and the decision problem TEMPORAL COLORING.

Definition 2.1. *Let (G, λ) be a temporal graph with lifetime T, where $G = (V, E)$. A* proper temporal coloring *of (G, λ) is a temporal coloring $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that every edge $e \in E$ is* temporally properly colored *in at least one time slot $t \in \lambda(e)$. The size of ϕ is the total number $|\phi| = |\bigcup_{i=1}^{T} \phi_i(V)|$ of colors used by ϕ.*

TEMPORAL COLORING

Input: A temporal graph (G, λ) with lifetime T and an integer $k \in \mathbb{N}$.
Question: Does there exist a proper temporal coloring $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ of (G, λ) using $|\phi| \leq k$ colors?

Note that TEMPORAL COLORING is a natural extension of the problem COLORING to temporal graphs. In particular, COLORING is the special case of TEMPORAL COLORING where the lifetime of the input temporal graph is $T = 1$, and therefore TEMPORAL COLORING is clearly NP-complete.

SLIDING-WINDOW TEMPORAL COLORING. In the definition of a proper temporal coloring given in Definition 2.1, we require that every edge is properly colored at least once during the whole lifetime T of the temporal graph (G, λ). However, in many real-world applications, where T is expected to be arbitrarily large, we may need to require that every edge is properly colored more often, and in particular, at least once during *every fixed period* Δ of time, regardless of how large the lifetime T is.

Before we proceed with the formal definition of this problem, we first present some needed terminology. For every time slot $t \in [1, T - \Delta + 1]$, the Δ-*time window* $W_t = [t, t + \Delta - 1]$ is the sequence of the Δ consecutive time slots $t, t + 1, \ldots, t + \Delta - 1$. Furthermore we denote by $E[W_t] = \bigcup_{i \in W_t} E_i$ the union of all edges appearing at least once in the Δ-time window W_t. We are now ready to introduce the notion of a *sliding* Δ-*window temporal coloring* and the decision problem SW-TEMP. COLORING.

Definition 2.2. *Let (G, λ) be a temporal graph with lifetime T, where $G = (V, E)$, and let $\Delta \leq T$. A* proper sliding Δ-window temporal coloring *of (G, λ) is a temporal coloring $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ such that, for every Δ-time window W_t and for every edge $e \in E[W_t]$, e is* temporally properly colored *in at least one time slot $t \in W_t$. The size of ϕ is the total number $|\phi| = |\bigcup_{i=1}^{T} \phi_i(V)|$ of colors used by ϕ.*

SLIDING WINDOW TEMPORAL COLORING (SW-TEMP. COLORING)

Input: A temporal graph (G, λ) with lifetime T, and two integers $k \in \mathbb{N}$ and $\Delta \leq T$.
Question: Does there exist a proper sliding Δ-window temporal coloring $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ of (G, λ) using $|\phi| \leq k$ colors?

Whenever the parameter Δ is a fixed constant, we will refer to the above problem as the Δ-SW-TEMP. COLORING (i.e. Δ is now a part of the problem name). Moreover, whenever both Δ and k are fixed constants, we will refer to the problem as the Δ-SW-TEMP. k-COLORING. Note that the problem TEMPORAL COLORING defined above in this section is the special case of SW-TEMP. COLORING where $\Delta = T$, i.e. where there is only one Δ-window in the whole temporal graph. Another special case of SW-TEMP. COLORING is the problem 1-SW-TEMP. COLORING, whose solution is obtained by iteratively solving the (static) COLORING problem on each of the T static instances of (G, λ). Thus 1-SW-TEMP. COLORING fails to fully capture the time dimension in temporal graphs; in the remainder of the paper we will assume that $\Delta \geq 2$.

Parameterized complexity. We use standard notation and terminology from parameterized complexity (Cygan et al. 2015). A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. We call the second component the *parameter* of the problem. A parameterized problem is *fixed-parameter tractable* (in the complexity class FPT) if there is an algorithm that solves each instance (I, r) in $f(r) \cdot |I|^{O(1)}$ time, for some computable function f. A parameterized problem L admits a *polynomial kernel* if there is a polynomial-time algorithm that transforms each instance (I, r) into an instance (I', r') such that $(I, r) \in L$ if and only if $(I', r') \in L$ and $|(I', r')| \leq r^{O(1)}$.

3 TEMPORAL COLORING

In this section we investigate the complexity of TEMPORAL COLORING. We start with our first hardness result.

Theorem 3.1. TEMPORAL COLORING *is NP-hard even if $k = 2$ and each snapshot has a clique and isolated vertices.*

One can also show that TEMPORAL COLORING remains hard even if each snapshot has very few edges.

Theorem 3.2. TEMPORAL COLORING *is NP-hard for all $k \geq 2$ even if each snapshot has $O(k^2)$ edges.*

Polynomial Kernel for TEMPORAL COLORING. We prove that, given a temporal graph (G, λ) for TEMPORAL COLORING with n vertices and T time slots, we can efficiently compute an equivalent instance (G', λ') with $T' \leq m$ time slots, where m is the number of edges in G. The main idea is that if we have sufficiently many time slots, every edge can be colored in its own time slot and any excess time slots can be removed (as they could be colored arbitrarily). Formally, Theorem 3.3 shows that TEMPORAL COLORING admits a polynomial kernel when parameterized by the number n of vertices.

Theorem 3.3. *Let (G, λ) be a temporal graph of lifetime T. Then there exists a temporal graph $(G', \lambda') = (G, \lambda)|_S$ for some $S \subseteq [T]$, $|S| \leq m = |E(G)|$ such that for any $k \geq 2$ we have that (G, λ) admits a proper temporal k-coloring if and only if (G', λ') admits a proper temporal k-coloring. Furthermore, (G', λ') can be constructed in $O(mT\sqrt{m + T})$ time.*

4 SW-TEMP. COLORING

In this section we thoroughly investigate the computational complexity of SW-TEMP. COLORING.

NP-Hardness. Before we present our main hardness result for SW-TEMP. COLORING, we start with the following intuitive observation.

Lemma 4.1. *For every fixed Δ, the problem $(\Delta + 1)$-SW-TEMP. k-COLORING is computationally at least as hard as the problem Δ-SW-TEMP. k-COLORING.*

The main idea is the following: given an algorithm A for $(\Delta + 1)$-SW-TEMP. k-COLORING, we can use A to also solve Δ-SW-TEMP. k-COLORING: we modify the instance of Δ-SW-TEMP. k-COLORING by inserting a trivial snapshot after every Δ consecutive snapshots, thus obtaining an equivalent instance of $(\Delta + 1)$-SW-TEMP. k-COLORING.

Since 1-SW-TEMP. k-COLORING is equivalent to solving T independent instances of static k-COLORING, Lemma 4.1 demonstrates that for any natural Δ, Δ-SW-TEMP. k-COLORING is at least as hard as k-COLORING. Thus, if k-COLORING is hard on some class $\mathcal{X}$ of static graphs, then Δ-SW-TEMP. k-COLORING is also hard for the class of always $\mathcal{X}$ temporal graphs.

Theorems 4.2 and 4.8 below imply that the converse is *not* true. In fact, there exist specific classes $\mathcal{X}$ of static graphs (graphs whose connected components have size $O(k)$ and graphs whose vertex cover has size $O(k)$, respectively) for which k-COLORING can be solved in *linear* time (for every fixed $k \geq 2$), although 2-SW-TEMP. k-COLORING is NP-hard on always $\mathcal{X}$ temporal graphs.

Theorem 4.2. *Let $k \geq 2$. Then 2-*SW-TEMP. k-COLORING *is NP-hard, even if $T = 3$ and:*

- *the underlying graph is $(k+1)$-colorable,*
- *the underlying graph has a maximum degree in $O(k)$, and*
- *every snapshot has connected components with size $O(k)$.*

Proof. We present a reduction from EXACT (3,4)-SAT (Tovey 1984) to 2-SW TEMP. 2-COLORING. The reduction can be easily modified to a larger number of colors, but we omit here the details. Recall that in EXACT (3,4)-SAT we are asked to decide whether a given Boolean formula ϕ is satisfiable and ϕ is in conjunctive normal form where every clause has exactly three distinct literals and every variable appears in exactly four clauses. Given a formula ϕ with n variables and m clauses, we construct a temporal graph (G, λ) consisting of three snaphots, which we will refer to as $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, and $G_3 = (V, E_3)$. We construct the following variable gadgets and clause gadgets. An illustration of the construction is given in Figure 1.

Variable gadget: For each variable x_i with $1 \leq i \leq n$ of ϕ we create five vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(2)}$, $v_{x_i}^{(3)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$. The vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(2)}$, and $v_{x_i}^{(3)}$ form a (not necessarily induced) P_3 in every snapshot, that is $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\} \in E_t$ and $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\} \in E_t$ for all $1 \leq t \leq 3$. Furthermore, we connect $v_{x_i}^{(1)}$ and $v_{x_i}^{(3)}$ in the second snapshot, that is, $\{v_{x_i}^{(1)}, v_{x_i}^{(3)}\} \in E_2$. Lastly, we create a full C_5 in snapshot three, that is, $\{v_{x_i}^{(3)}, v_{x_i}^{(4)}\} \in E_3$, $\{v_{x_i}^{(4)}, v_{x_i}^{(5)}\} \in E_3$, and $\{v_{x_i}^{(1)}, v_{x_i}^{(5)}\} \in E_3$.

Clause gadget: For each clause c_i with $1 \leq i \leq m$ of ϕ we create a total of 18 vertices. We create vertices $v_{c_i}^{(1)}$, $v_{c_i}^{(2)}$, and $v_{c_i}^{(3)}$ and connect them to a triangle in every snapshot, that is, $\{v_{c_i}^{(1)}, v_{c_i}^{(2)}\} \in E_t$, $\{v_{c_i}^{(2)}, v_{c_i}^{(3)}\} \in E_t$, and $\{v_{c_i}^{(1)}, v_{c_i}^{(3)}\} \in E_t$ for all $1 \leq t \leq 3$. In this proof, we refer to these vertices as the *core* of the clause gadget of clause c_i. Next, we add six vertices, which we refer to as the *extension* of the core of the clause gadget of clause c_i. Let these vertices be called $v_{c_i}^{(1,1)}$, $v_{c_i}^{(1,2)}$, $v_{c_i}^{(2,1)}$, $v_{c_i}^{(2,2)}$, $v_{c_i}^{(3,1)}$, and $v_{c_i}^{(3,2)}$. We connect $v_{c_i}^{(j,1)}$ and $v_{c_i}^{(j,2)}$ for all $1 \leq j \leq 3$ in every snapshot, that is, $\{v_{c_i}^{(j,1)}, v_{c_i}^{(j,2)}\} \in E_t$ for all $1 \leq j \leq 3$ and for all $1 \leq t \leq 3$. In the second snapshot, we connect the extension and the core in the following way.

- Edge $\{v_{c_i}^{(1,1)}, v_{c_i}^{(1,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(2)}, v_{c_i}^{(1)}\}$, that is, $\{v_{c_i}^{(2)}, v_{c_i}^{(1,2)}\} \in E_2$ and $\{v_{c_i}^{(1)}, v_{c_i}^{(1,1)}\} \in E_2$.
- Edge $\{v_{c_i}^{(2,1)}, v_{c_i}^{(2,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(2)}, v_{c_i}^{(3)}\}$, that is, $\{v_{c_i}^{(2)}, v_{c_i}^{(2,1)}\} \in E_2$ and $\{v_{c_i}^{(3)}, v_{c_i}^{(2,2)}\} \in E_2$.
- Edge $\{v_{c_i}^{(3,1)}, v_{c_i}^{(3,2)}\}$ forms a C_4 with edge $\{v_{c_i}^{(1)}, v_{c_i}^{(3)}\}$, that is, $\{v_{c_i}^{(1)}, v_{c_i}^{(3,2)}\} \in E_2$ and $\{v_{c_i}^{(3)}, v_{c_i}^{(3,1)}\} \in E_2$.

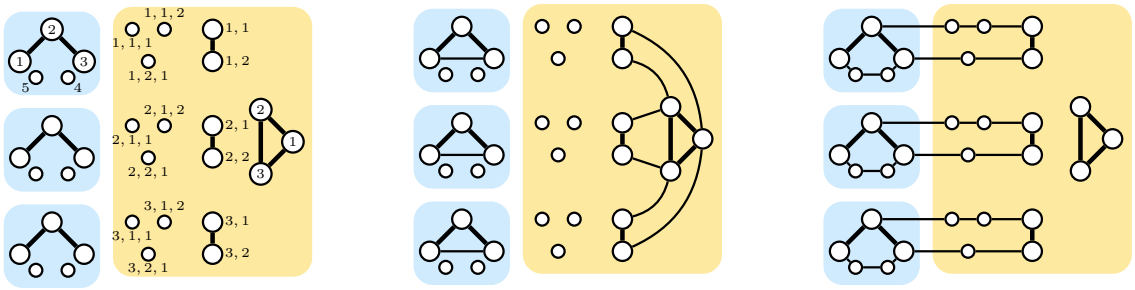
Lastly, we introduce nine auxiliary vertices that help to connect clause gadgets and variable gadgets. Let these vertices be called $v_{c_i}^{(j,1,1)}$, $v_{c_i}^{(j,1,2)}$, and $v_{c_i}^{(j,2,1)}$ for all $1 \leq j \leq 3$. In the third snapshot, we connect the extension of the core and these auxiliary vertices in the following way. For all $1 \leq j \leq 3$ we have that $\{v_{c_i}^{(j,1,1)}, v_{c_i}^{(j,1,2)}\} \in E_3$, $\{v_{c_i}^{(j,1,2)}, v_{c_i}^{(j,1)}\} \in E_3$, and $\{v_{c_i}^{(j,2,1)}, v_{c_i}^{(j,2)}\} \in E_3$.

Connection of variable and clause gadgets: The clause gadgets and variable gadgets are connected in the third snapshot. Let clause $c_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ with $1 \leq i \leq m$ have literals $\ell_{i,1}$, $\ell_{i,2}$, and $\ell_{i,3}$. Let $x_{i,j}$ with $1 \leq i \leq m$ and $1 \leq j \leq 3$ be the variable of the jth literal in clause c_i. If $\ell_{i,j} = x_{i,j}$, then $\{v_{x_{i,j}}^{(2)}, v_{c_i}^{(j,1,1)}\} \in E_3$ and $\{v_{x_{i,j}}^{(3)}, v_{c_i}^{(j,2,1)}\} \in E_3$. If $\ell_{i,j} = \neg x_{i,j}$, then $\{v_{x_{i,j}}^{(1)}, v_{c_i}^{(j,1,1)}\} \in E_3$ and $\{v_{x_{i,j}}^{(2)}, v_{c_i}^{(j,2,1)}\} \in E_3$.

This completes the construction. Recall that $\Delta = 2$ and $k = 2$. It is easy to check that the reduction can be computed in polynomial time. It remains to show that (G, λ) admits a proper sliding 2-window temporal 2-coloring if and only if ϕ is satisfiable.

($\Rightarrow$): Assume that we are given a satisfying assignment for ϕ. Then we construct a proper sliding 2-window temporal 2-coloring for G as follows. We start coloring the second snapshot and then show that we can color snapshots one and three in a way such that the complete coloring is a proper sliding 2-window temporal 2-coloring. If a variable x_i with $1 \leq i \leq n$ is set to true in the satisfying assignment, then we color the triangle of the corresponding variable gadget in a way that leaves only edge $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ monochromatic. To be specific, assume (for the remainder of this paragraph) we have colors yellow and blue, we color vertices $v_{x_i}^{(1)}$ and $v_{x_i}^{(2)}$ in yellow and vertices $v_{x_i}^{(3)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$ in blue. If variable x_i is set to false in the satisfying assignment, then we color the triangle of the corresponding variable gadget in a way that leaves edge $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\}$ monochromatic. To be specific, we color vertices $v_{x_i}^{(2)}$ and $v_{x_i}^{(3)}$ in yellow and vertices $v_{x_i}^{(1)}$, $v_{x_i}^{(4)}$, and $v_{x_i}^{(5)}$ in blue. For each clause c_i with $1 \leq i \leq m$ we choose one of its literals that satisfies the clause. Let the jth literal with $1 \leq j \leq 3$ be a satisfying literal of clause c_i for the given assignment. Then we color the core of the corresponding clause gadget in a way that leaves edge $\{v_{c_i}^{(j)}, v_{c_i}^{(j \bmod 3+1)}\}$ monochromatic. Note coloring the core uniquely determines how we have to color the extension of the core since the connecting edges are only present in the second snapshot and hence have to be properly colored. The auxiliary vertices can be colored arbitrarily.

Now we show how to color snapshot one. For each variable x_i with $1 \leq i \leq n$, we color $v_{x_i}^{(2)}$ in yellow and the remaining vertices of the corresponding gadget in blue. Note that this ensures that the edge which remains monochromatic in the second snapshot is properly colored in the first snapshot. For each clause c_i with $1 \leq i \leq m$ we color the core in a way that ensures that the edge which remains monochromatic in the second snapshot is properly colored in the first snapshot. We properly color all edges of the ex-



(a) Snapshot one. (b) Snapshot two. (c) Snapshot three.

Figure 1: Illustration of the reduction from EXACT (3,4)-SAT to 2-SW TEMP. 2-COLORING of the proof of Theorem 4.2. Vertices and edges in the yellow shaded areas (right) correspond to a clause gadget for clause $(x_1 \vee x_2 \vee x_3)$. Vertices and edges in the blue shaded areas (left) correspond to the variable gadgets for x_1, x_2, and x_3. Thick edges appear in every snapshot while thin edges only appear in one snapshot. In the first snapshot (a), the superscripts of the vertices used in the proof of Theorem 4.2 are shown. To keep the figure clean, those are omitted in the illustrations for snapshots two (b) and three (c).

tension and the auxiliary vertices arbitrarily. It is not hard to see that now the first Δ-window is properly colored.

Lastly, we show how to color the third snapshot. Note that for the variable gadgets, the coloring in snapshot two determines (up to switching the colors) how to color the variable gadgets in the third snapshot. This also determines how to color the auxiliary vertices and the extension of the core in the third snapshot. This potentially leaves edges of the extension monochromatic. Note that in the second snapshot, all extension edges are properly colored except the one which, in the third snapshot, is connected to a variable that, in the given assignment, satisfies the clause. It is straightforward to check that in this case, this particular extension edge is properly colored in the third snapshot. Lastly, the core is colored in a way that ensures that the edge that is colored monochromatic in the second snapshot is colored properly in the third snapshot. It is easy to check that now the second Δ-window is also properly colored.

($\Leftarrow$): Assume we are given a proper sliding 2-window temporal 2-coloring for (G, λ). Then we construct a satisfying assignment for ϕ in the following way: Note that in the second snapshot each variable gadget contains a triangle with exactly one monochromatic edge. The edge $\{v_{x_i}^{(1)}, v_{x_i}^{(3)}\}$ only exists in the second snapshot and hence is colored properly by any proper sliding 2-window temporal 2-coloring. This means that either edge $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ or edge $\{v_{x_i}^{(2)}, v_{x_i}^{(3)}\}$ is colored monochromatic. If $\{v_{x_i}^{(1)}, v_{x_i}^{(2)}\}$ is colored monochromatic then we set x_i to true, otherwise we set x_i to false. We claim that this yields a satisfying assignment for ϕ. Assume for contradiction that it is not. Then there is a clause c_j that is not satisfied. Without loss of generality, let x_1, x_2, and x_3 be the variables appearing in c_j. Then in the third snapshot, the clause gadget of c_j is connected to the variable gadgets of x_1, x_2, and x_3. It is easy to check that in any proper sliding 2-window temporal 2-coloring, exactly one edge of the extension of any clause gadget is colored monochromatic in the second snapshot, hence this is also the case in the clause gadget of c_j. Without loss of generality, let the monochromatically colored (in the second snapshot) extension edge of the clause gadget of c_j be connected to the variable gadget of x_1 in the third snapshot. It is easy to check that for the sliding 2-window temporal 2-coloring to be proper, the edge of the variable gadget of x_1 that is connected to the clause gadget of c_j in the third snapshot needs to be colored properly in the second snapshot. By construction of (G, λ) this is a contradiction to c_j not being satisfied by the constructed assignment. □

With small modifications to the reduction we get that SW-TEMP. COLORING remains hard under the following restrictions on the snapshots.

Corollary 4.3. SW-TEMP. COLORING *is NP-hard for all $k \geq 2$, $\Delta \geq c$, and $T \geq \Delta + 1$ for some constant c even if*
- *every snapshot is a cluster graph, or*
- *every snapshot has a dominating set of size one.*

The reduction presented in the proof of Theorem 4.2 also yields a running time lower bound assuming the Exponential Time Hypothesis (ETH) (Impagliazzo and Paturi 2001; Impagliazzo, Paturi, and Zane 2001).

Corollary 4.4. SW-TEMP. COLORING *does not admit a $k^{o(n)\cdot f(T+k)}$-time algorithm for any computable function f unless ETH fails.*

Proof. First, note that any 3SAT formula with m clauses can be transformed into an equisatisfiable EXACT (3,4)-SAT formula with $O(m)$ clauses (Tovey 1984). The reduction presented in the proof of Theorem 4.2 produces an instance of SW-TEMP. COLORING with $n = O(m)$ vertices, $k = 2$, and $T = 3$. Hence an algorithm for SW-TEMP. COLORING with running time $k^{o(n)\cdot f(T+k)}$ for some computable function f would imply the existence of an $2^{o(m)}$-time algorithm for 3SAT. This is a contradiction to ETH. □

Optimal Exponential-Time Algorithm Assuming ETH. In the following we give an exponential-time algorithm for Δ-SW-TEMP. COLORING that asymptotically matches the running time lower bound given in Corollary 4.4 assuming

the ETH. The main idea is to enumerate all partial proper sliding Δ-window temporal colorings for time windows of size 2Δ and then check whether we can combine them to a proper sliding Δ-window temporal coloring for the whole temporal graph.

Theorem 4.5. SW-TEMP. COLORING *can be solved in* $O(k^{4\Delta \cdot n} \cdot T)$ *time.*

Proof. For the sake of simplicity, we assume that T is divisible by Δ. The general case can be proven alike. We give the following algorithm for the problem:

1. For 2Δ-windows $W_i = [i\Delta + 1, (i+2)\Delta]$ for $i \in \{0, 1, \ldots, T/\Delta - 2\}$, enumerate all partial proper sliding Δ-window temporal colorings ϕ_{W_i}, where each trivial snapshot is colored in some fixed but arbitrary way[1].
2. Create a *directed acyclic graph* (DAG) with ϕ_{W_i} as vertices and connect ϕ_{W_i} and $\phi_{W_{i+1}}$ with a directed arc if the two proper Δ-temporal colorings agree on the overlapping part.
3. Create a source vertex s and connect it to all ϕ_{W_1} with a directed arc and we create a sink vertex t and add a directed arc from all $\phi_{W_{T/\Delta-2}}$ to it.
4. If there is a path from s to t, answer YES, otherwise NO.

The running time is dominated by checking whether s and t are connected in the last step of the algorithm. This can be done e.g. by a BFS on the constructed DAG. The DAG has at most $k^{2\Delta \cdot n} \cdot T$ vertices and at most $k^{4\Delta \cdot n} \cdot T$ edges. □

Fixed-Parameter Tractability. Next, we show how to extend the algorithm presented in Theorem 4.5 to achieve linear time fixed-parameter tractability with respect to the number n of vertices. The main idea is to reduce the number of non-trivial snapshots in each Δ-window.

Theorem 4.6. SW-TEMP. COLORING *can be solved in* $O(T)$ *time if* n *is a constant.*

Proof. We present a preprocessing step to reduce the number of non-trivial snapshots in any Δ-window and then use the algorithm of Theorem 4.5 to solve the problem.

The reduction rule is based on the observation that if some snapshot appears at least n^2 times in a Δ-window, then the edges of this snapshot can be properly colored with 2 colors within the Δ-window. In other words, all but n^2 copies of the snapshot in the Δ-window are redundant for optimal coloring and each of them could be replaced by the trivial snapshot. When implementing this idea one should take care to guarantee that replacing a snapshot by the trivial one does not reduce the number of copies of the snapshot in other Δ-windows which contain at most n^2 copies of the snapshot.

Formally the reduction rule is as follows. Since the number of different snapshots is at most $2^{\binom{n}{2}} \leq 2^{n^2}$, by the pigeonhole principle if $\Delta > 2 \cdot 2^{n^2} \cdot n^2$, then in every Δ-window there exists a snapshot that appears more than $2n^2$ times in that Δ-window. For every such a snapshot that contains at least one edge, we replace by the trivial snapshot one of its "middle" copies, that is, one that has at least n^2 copies appearing earlier and n^2 copies that appear later in the Δ-window. This reduction rule guarantees that every Δ-window that contains the modified snapshot also contains at least n^2 copies of the original snapshot appearing either earlier or later in the Δ-window.

The reduction rule can be applied exhaustively by linearly sweeping over all Δ-windows once in the following way. For each different graph (snapshot) we store a list of occurrences and update these lists every time we move the Δ-window by one. Having these lists, it is straightforward to count the occurrences and replace the middle ones by trivial snapshots. When we move the Δ-window, we just have to update two lists: the one of the graph that enters the Δ-window and the one of the graph that leaves. This requires a lookup table of size $2^{\binom{n}{2}} \leq 2^{n^2}$ but takes only linear time in T. Note that after this procedure, every Δ-window contains at most $2 \cdot 2^{n^2} \cdot n^2$ non-trivial snapshots.

Now we apply the algorithm of Theorem 4.5. Note that after the reduction step the number of *non-trivial* snapshots in every Δ-window depends only on n. Furthermore, since we can assume that $k \leq n$, the number of colorings that are enumerated in Step 1 of the algorithm in Theorem 4.5 is bounded by a function of n. This completes the proof. □

The FPT result of Theorem 4.6 is complemented by the following theorem, in which we exclude the possibility of a polynomial-sized kernel for SW-TEMP. COLORING with respect to the number n of vertices. This comes in contrast to the existence of a polynomial-sized kernel for TEMPORAL COLORING with respect to n (cf. Theorem 3.3).

Proposition 4.7. SW-TEMP. COLORING *does not admit a polynomial-sized kernel with respect to the number* n *of vertices for all* $\Delta \geq 2$ *and* $k \geq 2$ *unless NP* $\subseteq$ *coNP/poly.*

Structural Graph Parameters and Approximation. Finally, we investigate the possibility to structurally improve the fixed-parameter tractability result by replacing the parameter n with a smaller parameter. We answer this negatively by showing that Δ-SW-TEMP. k-COLORING remains NP-hard even if the underlying graph has a constant-size vertex cover, which is a fairly large structural parameter.

Theorem 4.8. *Let* $k \geq 2$*. Then 2-*SW-TEMP. k-COLORING *is NP-hard, even if the vertex cover number of the underlying graph is at most* $2k + 13$*.*

Next, we consider a canonical optimization version of SW-TEMP. COLORING, which we call MINIMUM SW-TEMP. COLORING, where the goal is to minimize the number of colors k. Using Theorem 4.6, we provide an FPT-approximation algorithm with an additive error of one where the parameter is the vertex cover number of the underlying graph. Considering that we cannot hope for an exact FPT algorithm for parameter "vertex cover number of the underlying graph" unless P = NP (cf. Theorem 4.8), this is the best we can get from a classification standpoint.

Theorem 4.9. MINIMUM SW-TEMP. COLORING *admits a linear time FPT-approximation algorithm with an additive error of one when parameterized by the vertex cover number of the underlying graph.*

[1] This is an important trick that allows us to use this algorithm for the FPT result in Theorem 4.6.

References

Aaron, E.; Krizanc, D.; and Meyerson, E. 2014. DMVP: foremost waypoint coverage of time-varying graphs. In *WG 2014*, 29–41.

Akrida, E. C.; Gasieniec, L.; Mertzios, G. B.; and Spirakis, P. G. 2016. Ephemeral networks with random availability of links: The case of fast networks. *J. Parallel Distrib. Comput.* 87:109–120.

Akrida, E. C.; Gasieniec, L.; Mertzios, G. B.; and Spirakis, P. G. 2017. The complexity of optimal design of temporally connected graphs. *Theory Comput. Syst.* 61(3):907–944.

Akrida, E. C.; Mertzios, G. B.; Spirakis, P. G.; and Zamaraev, V. 2018. Temporal vertex covers and sliding time windows. In *ICALP 2018*, 148:1–148:14.

Bentert, M.; Himmel, A.-S.; Molter, H.; Morik, M.; Niedermeier, R.; and Saitenmacher, R. 2018. Listing all maximal k-plexes in temporal graphs. In *ASONAM 2018*, 41–46.

Bui-Xuan, B.-M.; Ferreira, A.; and Jarry, A. 2003. Computing shortest, fastest, and foremost journeys in dynamic networks. *Int. J. Found. Comput. Sci.* 14(2):267–285.

Casteigts, A., and Flocchini, P. 2013a. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada.

Casteigts, A., and Flocchini, P. 2013b. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. Technical report, Defence R&D Canada.

Casteigts, A.; Flocchini, P.; Quattrociocchi, W.; and Santoro, N. 2012. Time-varying graphs and dynamic networks. *IJPEDS* 27(5):387–408.

Chen, J.; Molter, H.; Sorge, M.; and Suchý, O. 2018. Cluster editing in multi-layer and temporal graphs. In *ISAAC 2018*. To appear.

Clementi, A. E. F.; Macci, C.; Monti, A.; Pasquale, F.; and Silvestri, R. 2010. Flooding time of edge-markovian evolving graphs. *SIAM J. Discr. Math.* 24(4):1694–1712.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Enright, J.; Meeks, K.; Mertzios, G. B.; and Zamaraev, V. 2018. Deleting edges to restrict the size of an epidemic in temporal networks. *CoRR* abs/1805.06836.

Erlebach, T.; Hoffmann, M.; and Kammer, F. 2015. On temporal graph exploration. In *ICALP 2015*, 444–455.

Ferreira, A. 2004. Building a reference combinatorial model for MANETs. *IEEE Network* 18(5):24–29.

Flocchini, P.; Mans, B.; and Santoro, N. 2009. Exploration of periodically varying graphs. In *ISAAC 2009*, 534–543.

Fluschnik, T.; Molter, H.; Niedermeier, R.; and Zschoche, P. 2018. Temporal graph classes: A view through temporal separators. In *WG 2018*, 216–227.

Ghosal, S., and Ghosh, S. C. 2015. Channel assignment in mobile networks based on geometric prediction and random coloring. In *LCN 2015*, 237–240.

Giakkoupis, G.; Sauerwald, T.; and Stauffer, A. 2014. Randomized rumor spreading in dynamic graphs. In *ICALP 2014*, 495–507.

Himmel, A.; Molter, H.; Niedermeier, R.; and Sorge, M. 2017. Adapting the bron-kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Netw. Analys. Mining* 7(1):35:1–35:16.

Holme, P., and Saramäki, J., eds. 2013. *Temporal Networks*. Springer.

Impagliazzo, R., and Paturi, R. 2001. On the complexity of k-sat. *J. Comput. Sys. Sci.* 62(2):367–375.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity? *J. Comput. Sys. Sci.* 63(4):512–530.

Kempe, D.; Kleinberg, J.; and Kumar, A. 2002. Connectivity and inference problems for temporal networks. *J. Comput. Sys. Sci.* 64(4):820–842.

Latapy, M.; Viard, T.; and Magnien, C. 2018. Stream graphs and link streams for the modeling of interactions over time. *Social Netw. Analys. Mining* 8(1):61:1–61:29.

Leskovec, J.; Kleinberg, J. M.; and Faloutsos, C. 2007. Graph evolution: Densification and shrinking diameters. *TKDD* 1(1).

Mertzios, G. B.; Michail, O.; Chatzigiannakis, I.; and Spirakis, P. G. 2013. Temporal network optimization subject to connectivity constraints. In *ICALP 2013*, 657–668.

Mertzios, G. B.; Molter, H.; and Zamaraev, V. 2018. Sliding window temporal graph coloring. *CoRR* abs/1811.04753.

Michail, O., and Spirakis, P. G. 2016. Traveling salesman problems in temporal graphs. *Theor. Comput. Sci.* 634:1–23.

Michail, O., and Spirakis, P. G. 2018. Elements of the theory of dynamic networks. *Comm. ACM* 61(2):72–72.

Michail, O. 2016. An introduction to temporal graphs: An algorithmic perspective. *Internet Math.* 12(4):239–280.

Tang, J. K.; Musolesi, M.; Mascolo, C.; and Latora, V. 2010. Characterising temporal distance and reachability in mobile and online social networks. *Computer Communication Review* 40(1):118–124.

Tovey, C. A. 1984. A simplified NP-complete satisfiability problem. *Discrete Appl. Math.* 8(1):85–89.

Viard, T.; Latapy, M.; and Magnien, C. 2016. Computing maximal cliques in link streams. *Theor. Comput. Sci.* 609:245–252.

Yu, F.; Bar-Noy, A.; Basu, P.; and Ramanathan, R. 2013. Algorithms for channel assignment in mobile wireless networks using temporal coloring. In *MSWiM 2013*, 49–58.

Zschoche, P.; Fluschnik, T.; Molter, H.; and Niedermeier, R. 2018. The complexity of finding small separators in temporal graphs. In *MFCS 2018*, 45:1–45:17.